User Manual

Oclarity

Version 2.5

Copyright Notice
© 2004-2013 EmPowerTec AG, Taubenweg 20, 85238 Petershausen, Germany.

All rights reserved. This product and related documentation are protected by copyright and are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of EmPowerTec AG, and its licensors, if any.

Third Party Website Reference
EmPowerTec AG is not responsible for the availability of third-party Web sites mentioned in this document. EmPowerTec AG does not endorse and is not responsible or liable for any content, advertising, products, or other material on or available from such sites or resources. EmPowerTec AG will not be responsible or liable for any damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through any such sites or resources.

Trademarks
EmPowerTec is a trademark of EmPowerTec AG.
Other brands and their products are trademarks of their respective holders and should be noted as such.

1 Introduction 4
2 Installation 5
2.1 Requirements 5
2.2 Installation 5
2.3 Updates 6
2.4 Uninstalling 6
2.5 Printing this manual 6
2.6 Type mappings 6
2.7 Examples 7
3 GUI Basics 8
3.1 Overview 8
3.2 Arranging Screen elements 10
3.3 The Quick Access Toolbar 11
4 Understanding Oclarity projects 12
4.1 Creating a new project 12
4.2 Adding an XMI file to a project 12
4.3 Removing an XMI file from a project 12
4.4 Adding an existing OCL file to a project 13
4.5 Adding a new OCL file 13
4.6 Removing an OCL file from a project 13
4.7 Organizing your projects 13
5 Using the OCL editor 14
5.1 Introduction 14
5.2 Editing OCL files 14
5.3 Error highlighting 14
5.4 Intellisense 15
5.4.1 Complete Word 15
5.4.2 List members 15
5.4.3 Parameter Info 16
6 Checking OCL files 17
6.1 Single file check 17
6.2 Checking all files in the project 17
6.3 Package handling 17
6.3.1 Package names with spaces 17
6.4 States 18
6.5 Instantiated types 18

1 Introduction

OCL is a formal language intended to phrase expressions in object models, in particular in UML models. The purpose of OCL is to add precision to a model and complement the better known UML diagrams and use cases. If used properly, adding OCL expressions to a UML model can significantly increase the precision and ultimately the value of a UML model. Furthermore, the communication between the persons working on a software project is improved because all business logic described with OCL is available for all contributors to your project at any time.

OCL expressions can be used with varying intentions. Best known is the use as 'constraints'. Constraints state conditions that must be true in certain points of time.
Other applications of OCL expressions are initialization expressions for attributes and associations, derivation rules for additional auxiliary attributes and methods and the language independent description of method implementations.

OCL is language independent and thus helps you to specify more knowledge at the abstract level of the UML model than without using OCL. Instead of burying the business logic in complex programming language statements in software implementation files, it is stored at the heart of your software system: in your UML models.

It is beyond the scope of this document to provide an introduction to OCL. On our website http://www.empowertec.de, you can find links to OCL-resources.

Most UML modeling tools do not provide significant support for OCL. Oclarity adds comprehensive support for adding OCL expressions to a UML model:

- Syntax highlighting editor
- Full syntactic and semantic checking of OCL expressions ensures consistency with the model.
- Capability, to check all OCL expressions in a model at once. This is particularly useful if properties of a model are changed, e.g. class- or attribute names, method signatures and so on.
- Supports multiple modeling tools via XMI import.
- Tree like view of the underlying model and the OCL expressions.

2 Installation

2.1 Requirements

Oclarity requires:

- Microsoft Windows XP or higher
- Microsoft .NET framework version 4.0 Client Profile (or the full .NET framework 4.0)

If the .NET framework 4.0 Client Profile is not installed on your computer you can download and install it from Microsoft's website (http://www.microsoft.com/download/en/details.aspx?id=24872). You should choose the appropriate language.
The installation requires administrator privileges.

Due to the large variety in XMI formats, Oclarity only supports XMI files from selected tools. The list of supported files may increase in the future. Please let us know if you would like to use an XMI format that is not yet supported by Oclarity. The more votes a UML tool gets the higher is the probability that we will add support for the tools XMI format.

Currently Oclarity supports XMI formats from these tools:

Tool	XMI Version
Enterprise Architect 7.5, 8.0, 9.1	2.1
MagicDraw 15.0, 16.9, 17.0	Export to UML XMI 2.1 File
UML 2.5 Beta 1 Metamodel	

2.2 Installation

Load the current version of Oclarity from our website using this link:

http://empowertec.de/assets/Uploads/Oclarity/OclaritySetup.exe

Then execute the file.

If desired, the suggested destination directory can be changed. The installation requires administrator privileges.

2.3 Updates

To install a new version simply download it from our website, uninstall the old version and install the new version as described in chapter 2.2.

2.4 Uninstalling

To uninstall this software you may simply choose the according menu entry in the start menu.
Alternatively, you can use the Windows control panel to remove the software.

2.5 Printing this manual

If you print the manual with Adobe Acrobat Reader with default print settings, the pages are smaller than intended because Acrobat Reader scales the pages. To get a printout in original size, please uncheck all the scaling options in the Copies and Adjustments section of the Acrobat Reader print dialog box.

2.6 Type mappings

Often some type used in the model has the same semantics as an OCL builtin type but is syntactically different. For example, 'int' could be treated like 'Integer'. However, Oclarity does not know about this equivalence and thus treats 'int' as unknown type.
In such a case, a *type mapping* may be used. A type mapping is a pair of strings where the first string is the name of the type in the model and the second string is the name of the OCL builtin type.
Such a type mapping is created by adding an according entry to Oclaritys configuration file.
After installation, this file is called oclarity.etcfg. This file is passed as parameter to the main program EmPowerTec.Application.Shell.
Here are some sample entries:

```
<ApplicationPlugin name="Oclarity"
                   assembly="EmPowerTec.Ocl.Oclarity.dll"
                   class="EmPowerTec.Ocl.Oclarity.OclarityPlugin">
    <Option name="typemapping" value="int=Integer"/>
    <Option name="typemapping" value="Int=Integer"/>
    <Option name="typemapping" value="string=String"/>
    <Option name="typemapping" value="boolean=Boolean"/>
    <Option name="typemapping" value="double=Real"/>
</ApplicationPlugin>
```

In the default installation, this file is found here:
C:\Program Files\EmPowerTec AG\Oclarity\oclarity.etcfg

2.7 Examples

The Oclarity distribution comes with various example projects. The related project files are stored in the subfolder "Examples" of the Oclarity installation directory.

3 GUI Basics

3.1 Overview

Oclarity uses the concept of the "Ribbon control" that was introduced by Microsoft with their Office 2007 product line. The idea of the Ribbon control is to organize related functionality in dedicated "Ribbon tabs" and to selectively activate the Ribbon tab that is relevant to the current context.

Oclarity has two ribbon tabs:

The "Project" Ribbon tab is used to modify an Oclarity project (see below for an introduction to the concept of an Oclarity project)

The "Edit" Ribbon tab is used to support editing OCL files.

Oclarity tries to activate the appropriate tab whenever the user clicks on a given element. It is also possible to activate an inactive Ribbon tab by clicking on its "handle".

The Ribbon tab is located at the top of the main window like a traditional menu.

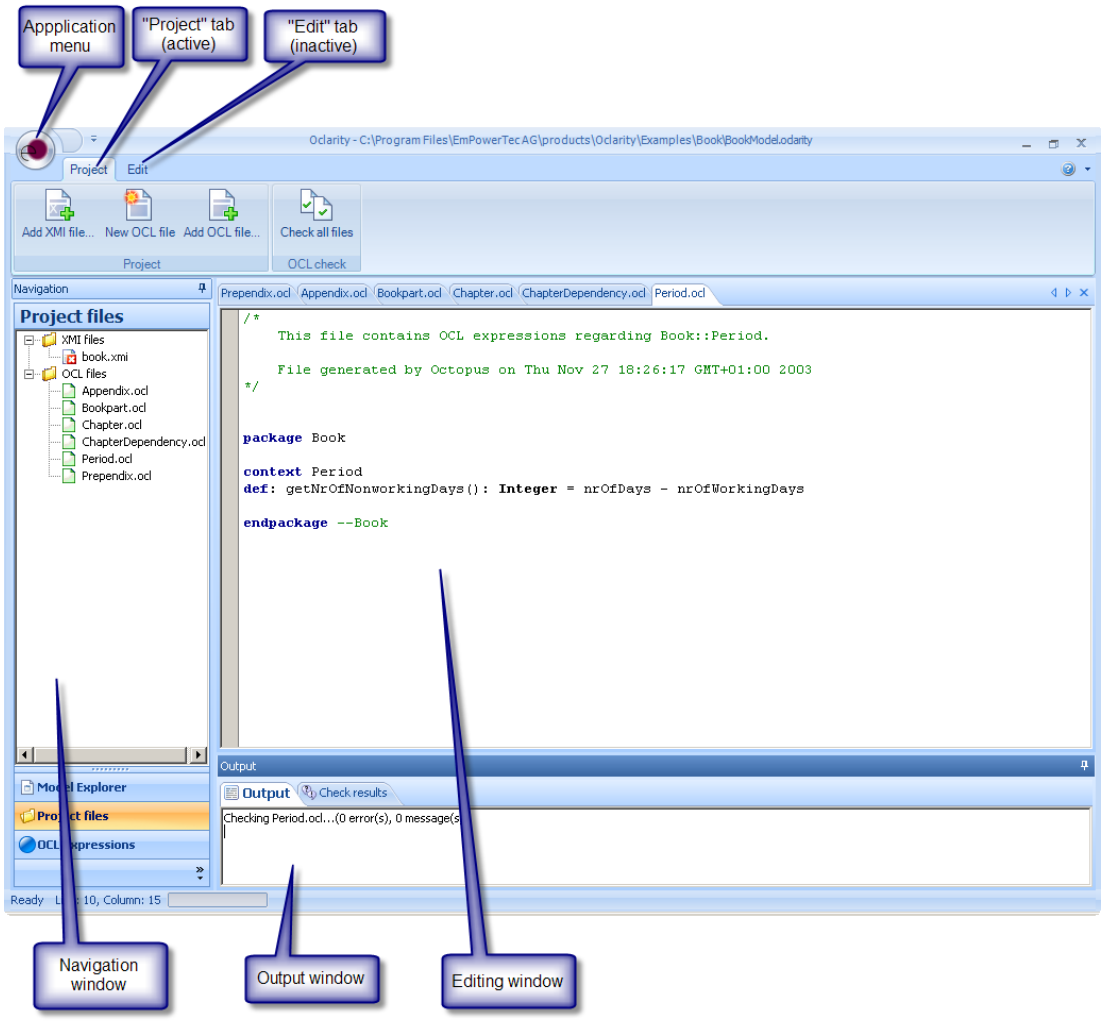

Figure 1: GUI overview

The left side of the main window – the Navigation window – contains a set of views that can be used to manipulate and view various aspects of the projects:

- The files that are part of the project
- The packages and classes of the model (if at least one XMI file has been added to the project)
- An overview of the OCL expressions grouped according to their context and expression type (requires a prior check of at least one OCL file)

Only one of these views is active at a given time. Click on a views tab to activate the view.

The lower right area of the main window contains the output window. This window contains two tabs:

- The "Output" tab contains information about the progress and status of an OCL check or other relevant messages, e.g. errors or warnings during the loading of an XMI file.
- The "Check result" tab contains a tabular list of error which can be used to easily locate the error in the related OCL file. If no errors have occurred during a check the error list in this tab is empty.

The rest of the main window is used as editing area for OCL files.

3.2 Arranging Screen elements

The "Navigation" area and the "Output" area can be attached to an arbitrary side of the main window by clicking in their title bar and dragging the area to the desired position.
A visual indicator appears that signals where the area will be placed.

The following screenshot shows the "Navigation" area docked at the right border. The "Output" area is currently moved and would be docked at the upper border if the mouse button would be released.

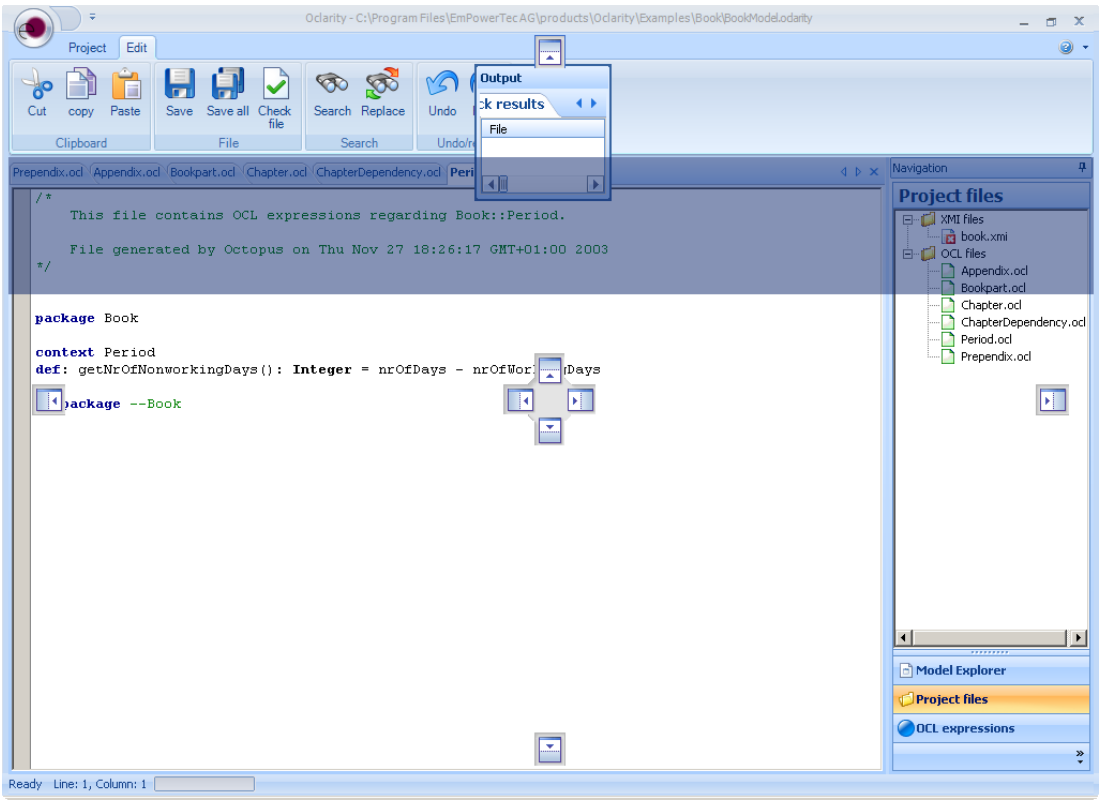

Figure 2: Arranging GUI elements

In a similar way, multiple editing windows can be created. For example, in the following screenshot two editing windows are used.

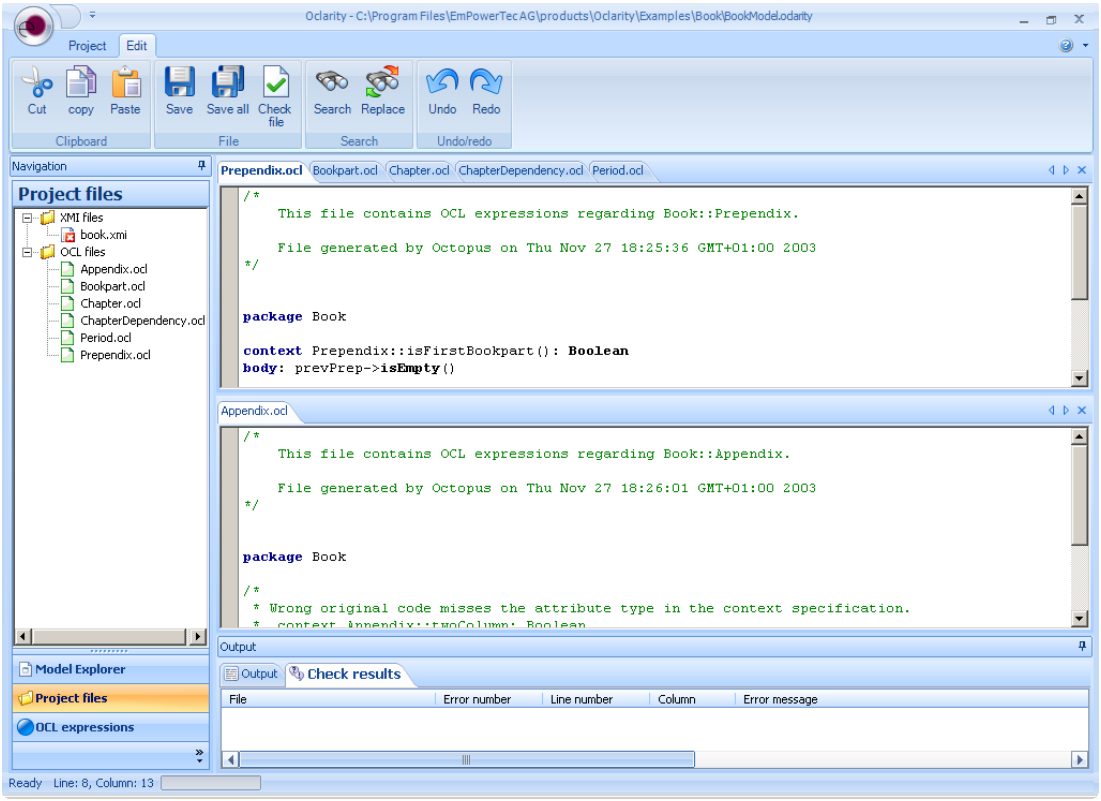

Figure 3: multiple editing window

3.3 The Quick Access Toolbar

At the right side of the "Application button" there is a small handle that can be used to access the Ribbons "Quick Access Toolbar" (QAT).

Most interface element on a Ribbon tab can be added to the QAT.
To add or remove command to or from the QAT click on the small handle at the right side of the QAT and click on the menu entry "Customize Quick Access Toolbar…" .

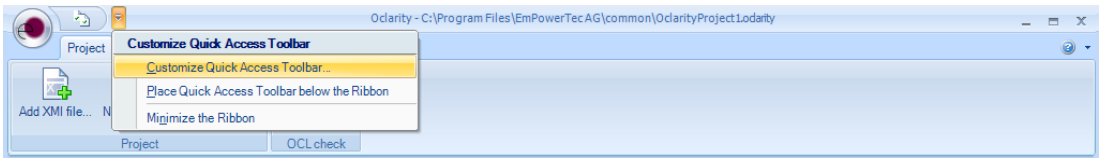

Figure 4: Modifying the Quick Access Toolbar

4 Understanding Oclarity projects

Oclarity reads a UML model from one or more XMI file(s). All OCL expressions are validated against the model defined in these XMI files.
OCL expressions are stored in text files. An Oclarity project groups a single XMI file and a set of OCL files together. Such a project is stored as text file on your computer. You can create as many different projects as you like.
The Oclarity project file, the XMI files and the OCL files remain distinct files.

4.1 Creating a new project

Use the application menus entry "New project" to create a new project. Oclarity will give the project a default name. You can use the application menus entry "Save project as…" to save the project under another name and/or directory.

4.2 Adding an XMI file to a project

Before OCL files can be checked, one or more XMI files have to be added to the project.

To add an XMI file, the "Project" tab of the Ribbon must be activated. Then click on the button labeled "Add XMI file…".
A dialog appears that lets you choose a XMI file.

After the XMI file has been added, you can inspect the "Model Explorer" view to see Oclarities view of the model defined in the XMI file.

Multiple XMI files can be added to a project.

It should be noted that Oclarity does not handle all elements contained in an XMI file. This may lead to errors or warnings in the output window.
In addition, the model could have inconsistencies which lead to inconsistencies in the XMI file.
In general, Oclarity tries to extract as much information as possible from the model and just reports errors or inconsistencies.

4.3 Removing an XMI file from a project

Removing an XMI file from the project does not delete the actual file; it just removes the file from the project.

Make sure that the "Project Explorer" view is active.
Right click on the XMI file that you want to remove.
A context menu appears. Select the "remove" entry.
You will be asked for a confirmation.

After the XMI file has been removed you should reload the project (Oclarity asks you to reload the project automatically).

4.4 Adding an existing OCL file to a project

Make sure that the "Project" tab of the ribbon control is activated, either by clicking on its handle or by activating and clicking in the "Project Explorer" view.

To add an existing OCL file, the "Project" tab of the Ribbon must be activated. Then click on the button labeled "Add OCL file…".
A dialog appears that lets you choose an OCL file.

4.5 Adding a new OCL file

Make sure that the "Project" tab of the ribbon control is activated, either by clicking on its handle or by activating and clicking in the "Project Explorer" view.

To add a new OCL file, the "Project" tab of the Ribbon must be activated. Then click on the button labeled "New OCL file".

A dialog will appear that lets you choose a directory and filename for the new OCL file.

4.6 Removing an OCL file from a project

Removing an OCL file from the project does not delete the actual file; it just removes the file from the project.

Make sure that the "Project Explorer" view is active.
Right click on the OCL file that you want to remove.
A context menu appears. Select the "remove" entry.
You will be asked for a confirmation.

4.7 Organizing your projects

Also this is technically not necessary we recommend putting a project file along with its OCL and XMI files in the same directory. This way, it is easy to move a project on disk just by copying the whole directory. You may use subdirectories but the folder hierarchy is not reflected in the Project Explorer.

5 Using the OCL editor

5.1 Introduction

The OCL code editor offers powerful features for editing OCL code. This chapter introduces the various features.

You can activate the editor either by clicking on an edit window or by double clicking on an OCL file in the "Project Explorer" view.

5.2 Editing OCL files

The "Edit" tab of the Ribbon provides the standard editing functionality.
The button "Check file" in the "File" group can be clicked to check the OCL file that is currently edited.

5.3 Error highlighting

The editor highlights errors as you type. After a delay in typing of 1 second the code is checked and any errors are highlighted. If you move the mouse cursor above a highlighted code fragment, the error message is displayed in a tooltip.

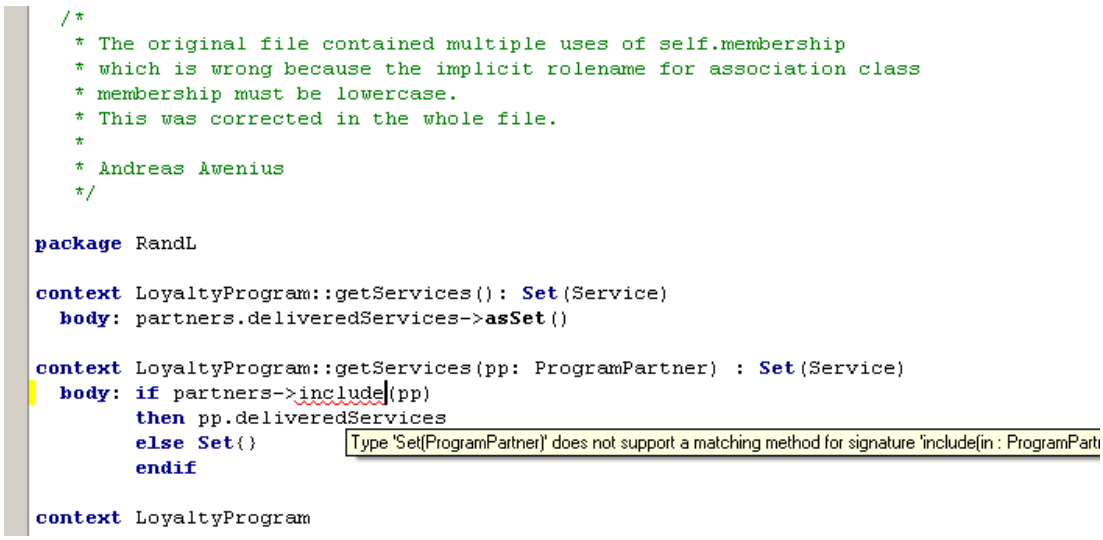
/*
* The original file contained multiple uses of self.membership
* which is wrong because the implicit rolename for association class
* membership must be lowercase.
* This was corrected in the whole file.
*
* Andreas Awenius
*/
package RandL
context LoyaltyProgram::getServices(): Set(Service)
body: partners.deliveredServices->asSet()
context LoyaltyProgram::getServices(pp: ProgramPartner) : Set(Service)
body: if partners->include(pp)
then pp.deliveredServices
else Set{}
endif
context LoyaltyProgram
Type 'Set(ProgramPartner)' does not support a matching method for signature 'include(in : ProgramPart

Figure 5: error highlighting in the OCL editor

In the example above, the method should be "includes" instead of "include".

5.4 Intellisense

Oclarity supports some advanced editing features[1]:

- Complete Word
- List Members
- Parameter Info

Please note, that for technical reasons the Intellisense-Features do not support the complete OCL language but just the most important requirements.

5.4.1 Complete Word

If you type ctrl-j (ctrl-key and 'j' key at the same time) the OCL Editor will try to make an appropriate suggestion. The suggestions are dependent of the context. This feature can be used as part of an identifier or it can be used when the editing cursor is not directly placed after an identifier.

5.4.2 List members

When a dot-character ('.') or an arrow ('->') is typed, the OCL Editor tries to make an appropriate suggestion. This usually means displaying a list of attributes, associations or methods.
When two colons ('::') are typed, the OCL Editor tries to suggest sub packages, static member or enumeration values.

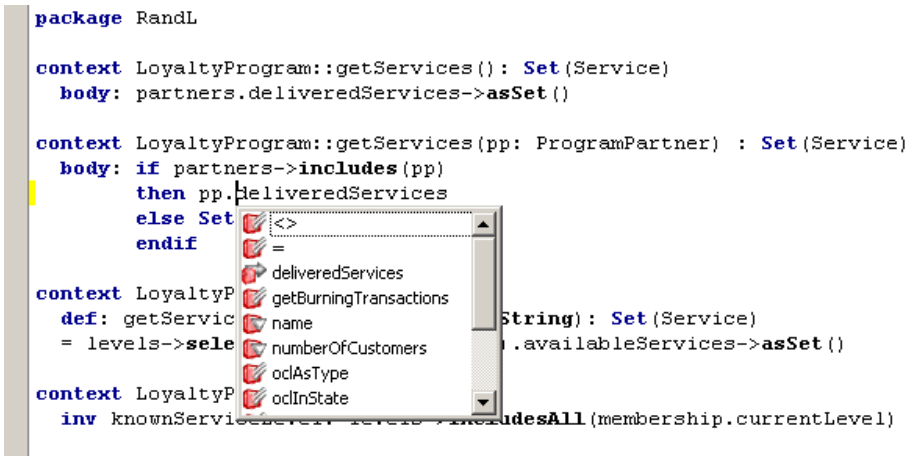

Figure 6: Member completion

[1] For technical reasons, the support of these features is limited to some standard constructs.

5.4.3 Parameter Info

If possible, the OCL Editor tries to display the signature of a method that is used and also tries to highlight the current parameter.

6 Checking OCL files

6.1 Single file check

While editing an OCL file you can click on the "Check file" button in the "Edit" tab of the Ribbon to check the current OCL file.

6.2 Checking all files in the project

At any time, you can click on the "Check all files" button in the "Project" tab to check all files defined in the project.

The "Output" tab of the Output window will display the status of the check and a short summary.

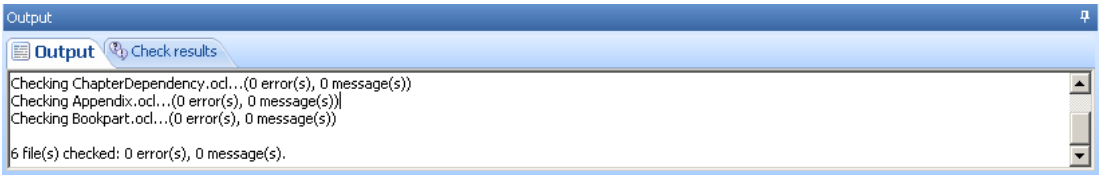

Figure 7: Output tab after OCL check

If errors have been detected during the check, they are displayed in a list in the "Check results" tab.

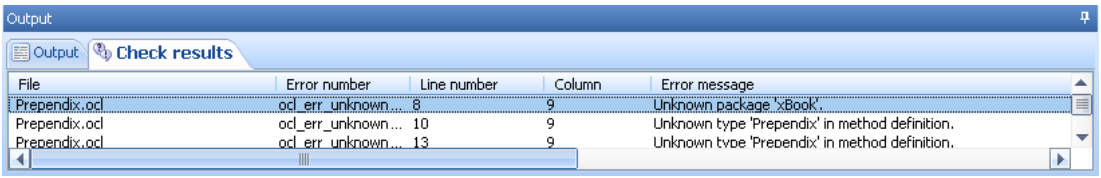

Figure 8: Check results tab after an OCL check

A Double click on an entry on this list positions the cursor on the offending token so you can immediately correct it.

6.3 Package handling

6.3.1 Package names with spaces

Most UML tools allow package names with spaces but this is not supported by the OCL grammar. As a workaround the spaces have to be "escaped" (preceded by a backslash character '\') in the OCL code.
For example, if you want to reference a type named 'Customer' in the package 'Logical View::Business Objects', the OCL code must use 'Business\ Objects::Customer.
It is not possible to mix underscores and spaces in package names. Our recommendation is not to use package names with spaces in your Rational Rose models.

6.4 States

In this version of Oclarity state machines are not suported (oclInState() cannot check for valid states).

6.5 Instantiated types

An instantiated type is a type that is not used on its own but only in combination with another type. This concept is called 'templates' in C++ or 'generics' in Java and C#. For example, a container class 'Set' may not be used on its own but only as container for specific contained elements, e.g. instances of class 'Service'.

In OCL, the syntax to express such an instantiation is

```
Type1(Type2)
```

e.g.

```
Set(Service)
```

Since this syntax deviates from the syntax of some widely used programming languages, we decided to allow both notations interchangeably.

This means, if you are modeling a PSM (Platform Specific Model) and thus use concrete types from your target language, this syntax can be used:

```
Type1<Type2>
```

e.g.

```
Set<Service>.
```

Thus your class definitions remain compatible with common programming languages.

If you are modeling a PIM (and thus remain independent from a concrete programming language) we recommend using the standard OCL syntax:

Type1(Type2)

e.g.

Set(Service)

The alternative spelling using ‘<’ and ‘>’ can be used anywhere, in particular for operation return types and operation parameter types.
In the OCL code however, the syntax must be used always as defined by the OCL grammar:

```
Set(Service)
```